Host-Oriented Security Test Suite (HOSTS)

James A. Finegan

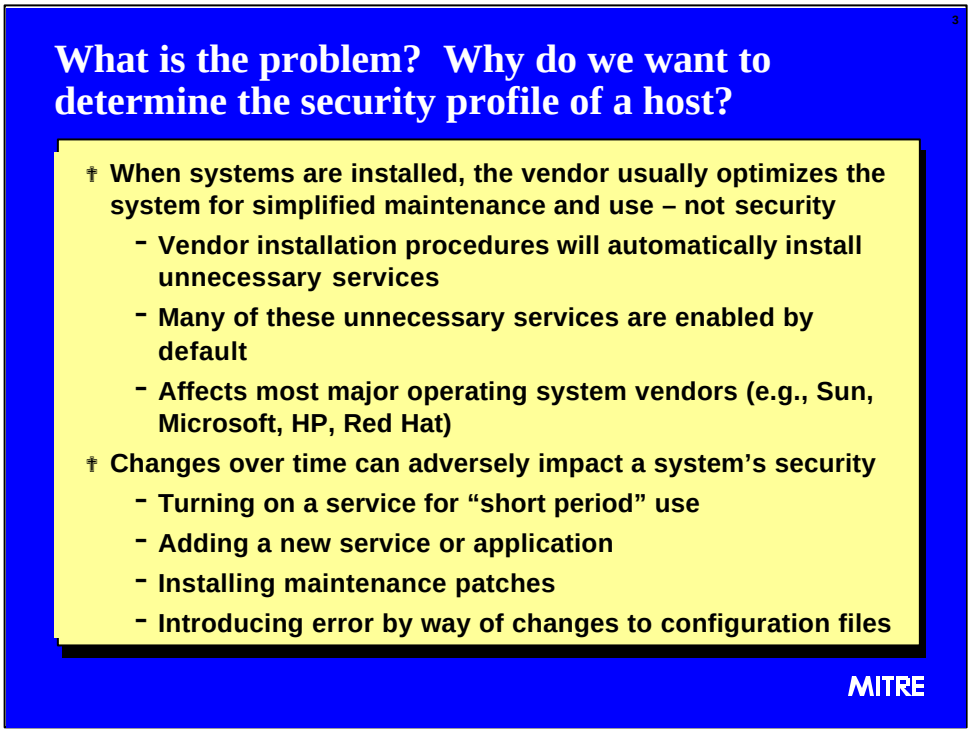
18th Annual Computer Security Applications Conference
Enterprise Security
December 12, 2002

MITRE

Agenda

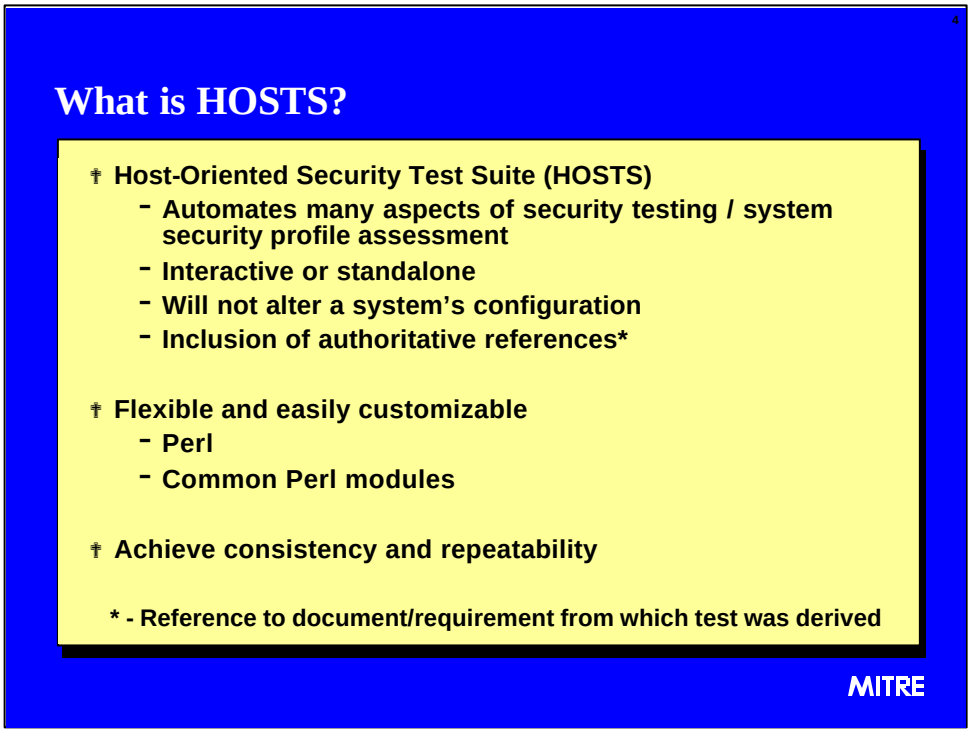
- Problem Description
- What is HOSTS?
  - What does HOSTS do?
  - Is HOSTS available?
  - The HOSTS Structure
  - The HOSTS Test Procedure
  - Visualization of HOSTS
  - Where do the HOSTS Test Series come from?
  - Existing Security Test Series
- Can HOSTS Really Make a Difference?
  - Sample HOSTS test series entries
  - The corresponding HOSTS test results
- Future - Where is HOSTS going
- Questions and Answers

MITRE



Major operating system vendors have taken the position that it is easier to enable services in a "one size fits all" model than to attempt to guide the user community through the process of enabling required services. This includes:

- HP's HP-UX (Tooltalk, "R" Services, SNMP, SMTP, NFS)
- Red Hat's Linux (NFS, VNC, Kudzu, SMTP, ICQ/chat tools)
- Microsoft's Windows (File and Print sharing, Posix and OS2 subsystems, IP forwarding, SNMP)
- Sun Microsystems' Solaris (Admind, Tooltalk, Web servers [AnswerBook, Apache], "R" Services, SNMP, SMTP, NFS)

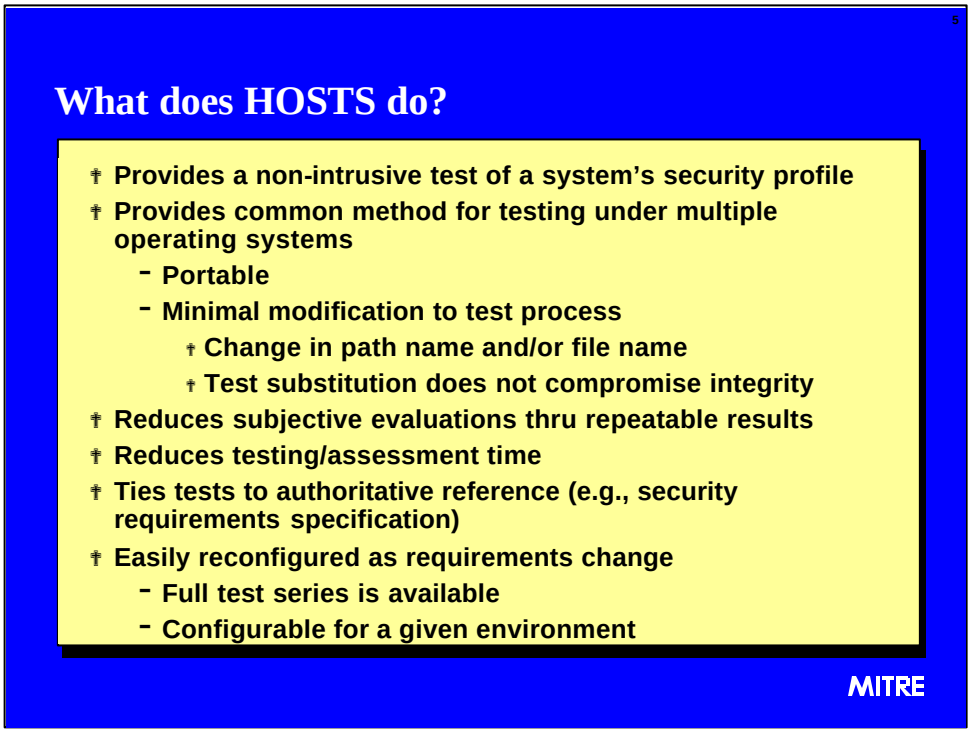


The **Host-Oriented Security Test Suite** (HOSTS) is a utility that automates many aspects of security testing performed within a host environment; it can be utilized either for interactive or standalone testing.

The utility is both flexible and easily customizable, requiring only the Perl 5 programming language and common Perl modules such as *FIND* and *ctime*. It has run under both Unix and Windows (via ActiveState Perl).

By using HOSTS as part of the security evaluation process, a level of consistency and repeatability in testing can be readily achieved.

- Additional benefit of a reduced probability for an operator-induced error, which can skew tests results, is also achieved.
- A direct consequence is the reduction for both the amount of time and level of effort required to perform security testing/profile assessment.



As new tests are developed, the existence of the original test series will facilitate both a historical perspective and an ability to perform full regression testing. It is also important that the developed tests can be quickly ported from one operating environment to another. Finally, as tests are ported, any substitutions made, particularly at a low level, do not compromise the integrity of the overall test.

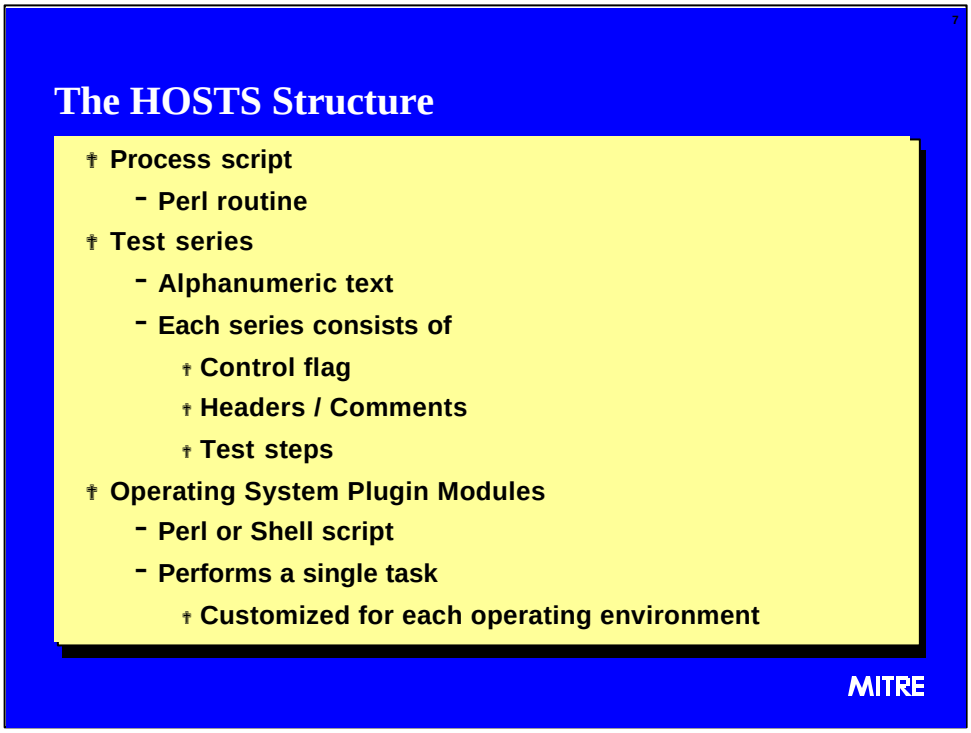
The need for unambiguous results led to a brief dilemma when it was realized that circumstances do occur where mutually exclusive conditions could exist, both of which represent an equally secure environment. For example, under some variants of Unix, the non-existence of a file is just as secure as when the file exists and a particular control variable has been specified.

A second situation arises when requirement differences occur when projects use the same baseline operating system as the initial foundation. For example, Application X may require an FTP server. Application Y does not. Consequently, a test to verify that the FTP daemon has been disabled, when running an Application X system, would fail since FTP must be enabled. The same test must pass when executed on an Application Y system. Once again we have a case of mutual exclusivity – a test that will pass in one or more configurations but fails in others.

As a result, the ability to bypass specific tests was added to handle situations such as mutual exclusivity and mission non-applicability.

At the point HOSTS was created, the original sponsor requested that the tool be created as Open Source. This was done in the hope that the world community would find the tool useful and feel free to openly contribute to a “growing” library of security test cases. This library then becomes available for use everywhere as a tool to help ensure a system is being securely maintained.

The lesser GNU license was chosen since HOSTS is meant to serve only as a foundation for this type of testing (e.g., a third party could add a proprietary wrapper around the tool).

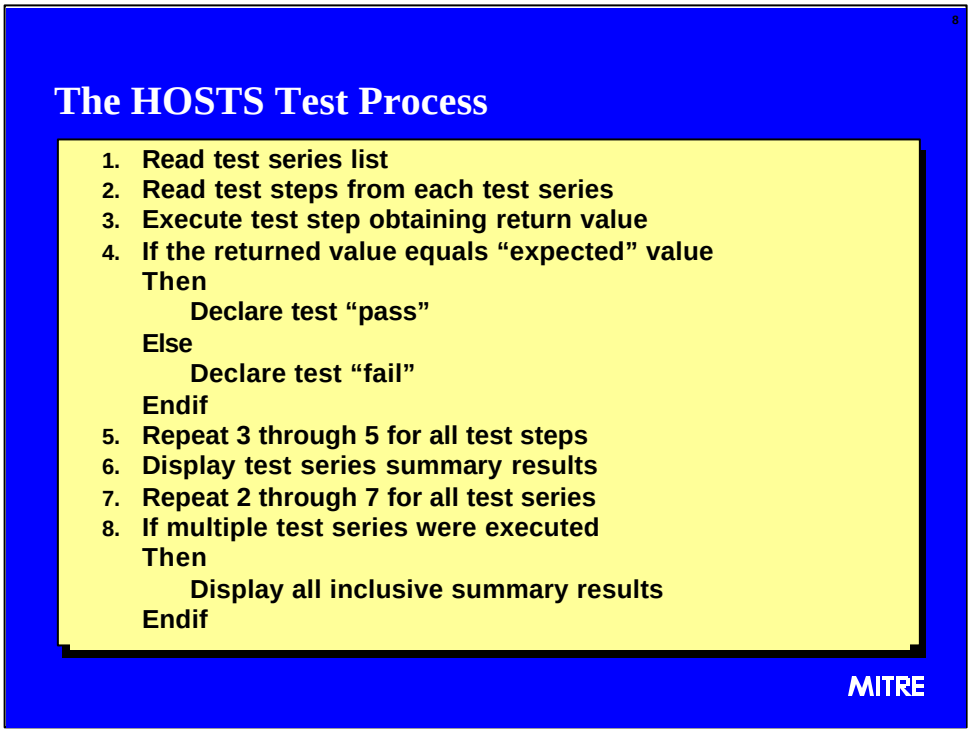


The hosts.pl script reads the test series files as input. Non-comment lines are interpreted and "executed" by calling the applicable plugin module. The returned result is then compared to expected results. When the results match, a pass is declared. Otherwise, the test fails. Following the completion of a test series (after the script has read in and processed all tests within the test series file), a summary is displayed.

The test series is a sequence of individual tests that utilize plugin modules to verify a configuration (e.g., is `/etc/default/login` set up correctly), a capability (are login failures recorded properly), or an application functionality (e.g., does the chmod command work as expected). Some steps may be included to set up the test environment before actual testing can commence. Others are used to clean up afterwards. Header and comment definitions are used to display descriptive messages as the test process executes.

Individual tests consist of a unique step identifier, a descriptive narration on the test, the calling sequence for the modular plugin, the expected return value(s), and two reference fields for cross-referencing the test step into requirements specifications.

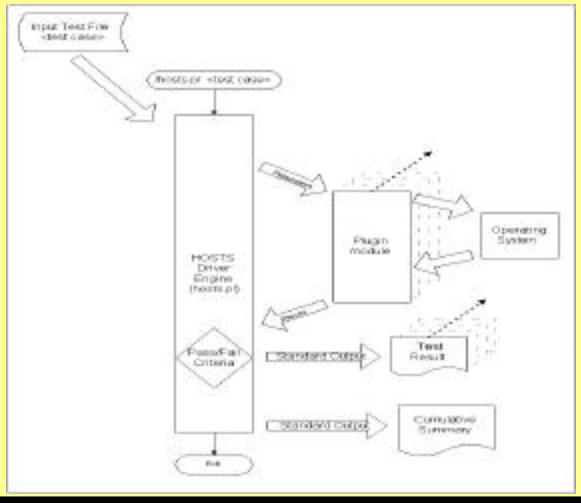
Plugin modules are tailored to the operating system being tested. They perform a single task returning an answer (e.g., "match" or "no match") followed, if applicable, by test data on which the result was based.



The HOSTS test process:

1. Read in a list of one or more test series to be processed. The list is entered as options on the command line. In no tests are listed, a default sequence will be used.
2. Read in the individual test steps (including comments and section headers) from each test series.
3. With the exception of those steps tagged to be skipped, each test step is executed sequentially.
4. If the returned value from a test step matches one of the allowed “expected” return values, a “pass” is declared for the test step. Otherwise, a “fail” is declared for the test step.
5. The process of executing tests steps and comparing returned values with expected return values is repeated until all executable tests within the active test series have been run.
6. A summary of the accumulated test results is displayed. The summary includes pass/fail percentages and a list of applicable requirements that correspond to the pass/fail results.
7. If more than one test series was specified via the command line (or in default list), the processing of individual tests series will continue until all series have been processed.
8. If applicable (e.g. more than one test series was processed), a global summary on the accumulated results for all series is displayed. This summary also includes global pass/fail percentages and a list of applicable requirements that correspond to the pass/fail results.

Visualization of HOSTS



MITRE

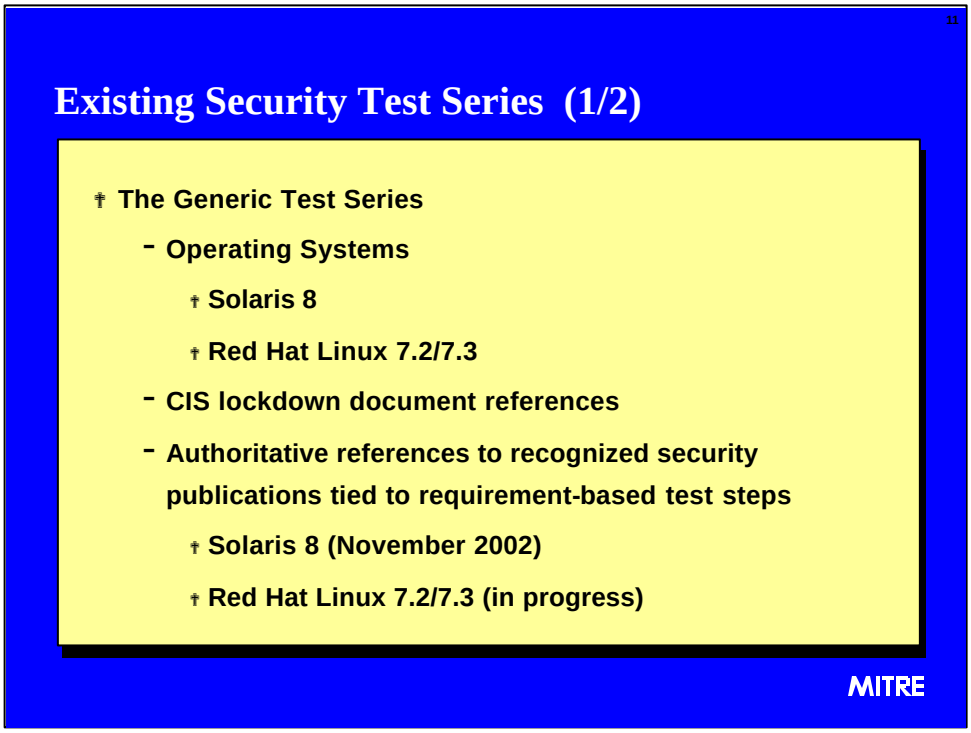
10

Where do the HOSTS Test Series come from?

The developed test series are based on “Best Practices”

- **Center for Internet Security**
- **SANS Lock-Down Guides**
- **Solaris and Red Hat Linux Security Guidelines**
- **Seasoned System Administrators**
- **Unix/Linux Security Books**
- **Unix Security Test Procedures**

MITRE

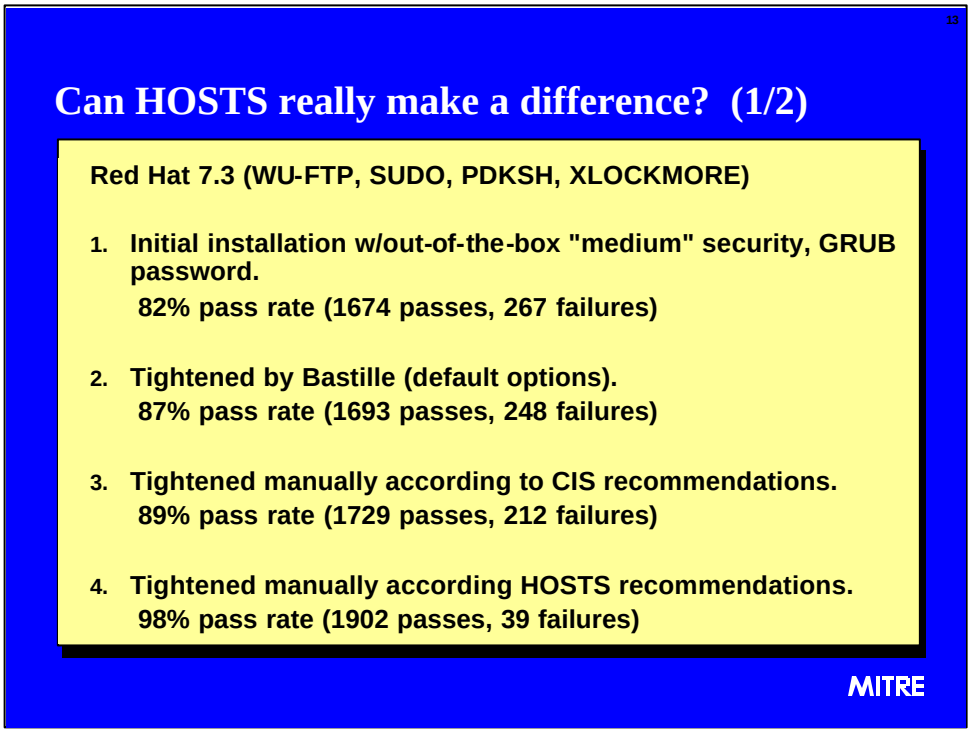


One of the most difficult hurdles to overcome has turned out to be the authoritative reference integration effort. While it turned out to be relatively easy to track most of the test steps back to an authoritative reference, less than 5% remain tagged as “To Be Specified (TBS)”.

Existing Security Test Series (2/2)

- Available Unix Test Series
  - Audit Series: Tests the usability, functionality and configuration of the audit daemon. (Linux series based on SNARE.)
  - DAC Series: Tests the usability, functionality and configuration of the Discretionary Access Control.
  - Encryption Series: Tests the rudimentary functionality of the Operating System supplied encryption logic.
  - I&A Series: Tests the usability, functionality and configuration of the Identification and Authentication capability.
  - Miscellaneous Series: Tests the functionality and configuration of miscellaneous aspects of the environment.
  - OS Series: Tests the functionality and configuration of the Operating System aspects of the environment.
  - Version Series: Tests for the presence of specific versions/releases for applications with known vulnerabilities.

MITRE

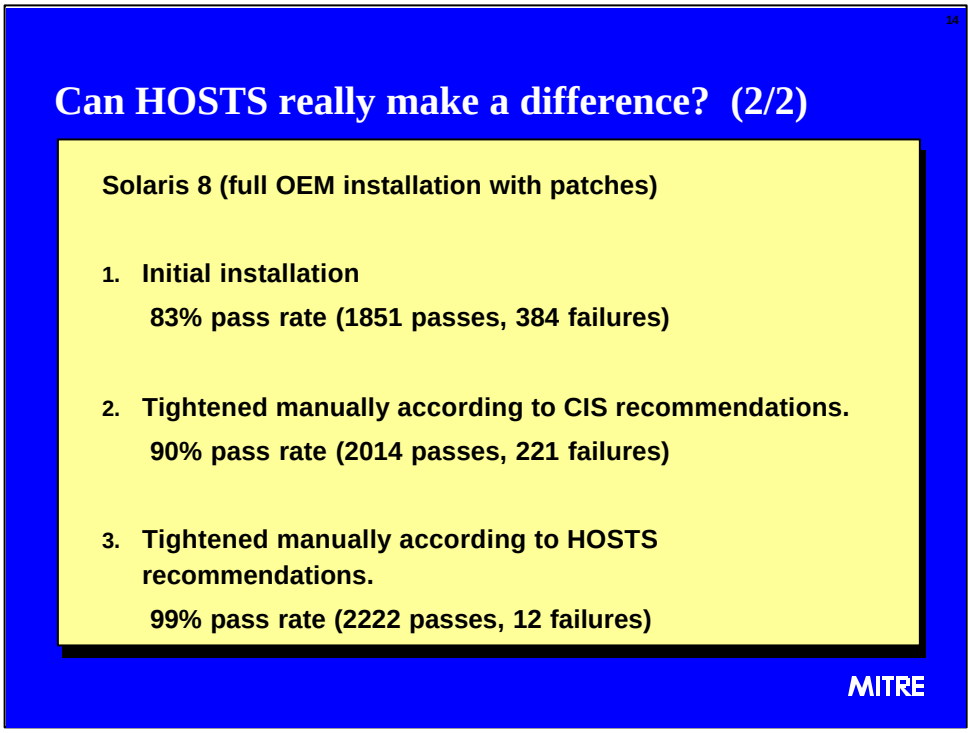


An improved score is clearly visible as the system was progressively locked down. It should be noted that this system could never achieve a 100% test pass rate. The following list details the primary causes for failures as they apply to the tested system:

- The tested system needed several services (e.g., NFS client, the GNOME window manager). Had it been possible to fully disable these services, a higher pass rate could have been achieved.
- A few of the failures are related to the inability of the operating system to pass some of the tests. For example, the POSIX useradd command allows an administrator to create an account that has the same UID as an existing account. Consequently, a test to verify that it is not possible to create an account with a duplicate UID will fail.
- A few of the failures are related to the approach used to access specific privilege elevating commands (e.g., ps and su) – particularly if the approaches happen to be mutually exclusive. (If there are multiple approaches to controlling access, test steps examining for approach A could fail if approach B happens to have been implemented.)

Caveat: As a system is tightened, it becomes “less usable” (e.g., services are being disabled). The key is to strike a balance where the system has been tightened to the fullest extent possible while at the same time mission required services are available. Extending this to the testing process means that a perfect score (e.g., 100% pass rate) is not really desirable, let alone achievable, and may reflect a state that is unusable to the system’s targeted user community.

For most commonly used applications (and services), configuration options exist that permit one to tighten the security profile of application. This facilitates the both the locking down of the application and its use – often invisibly to the user community.

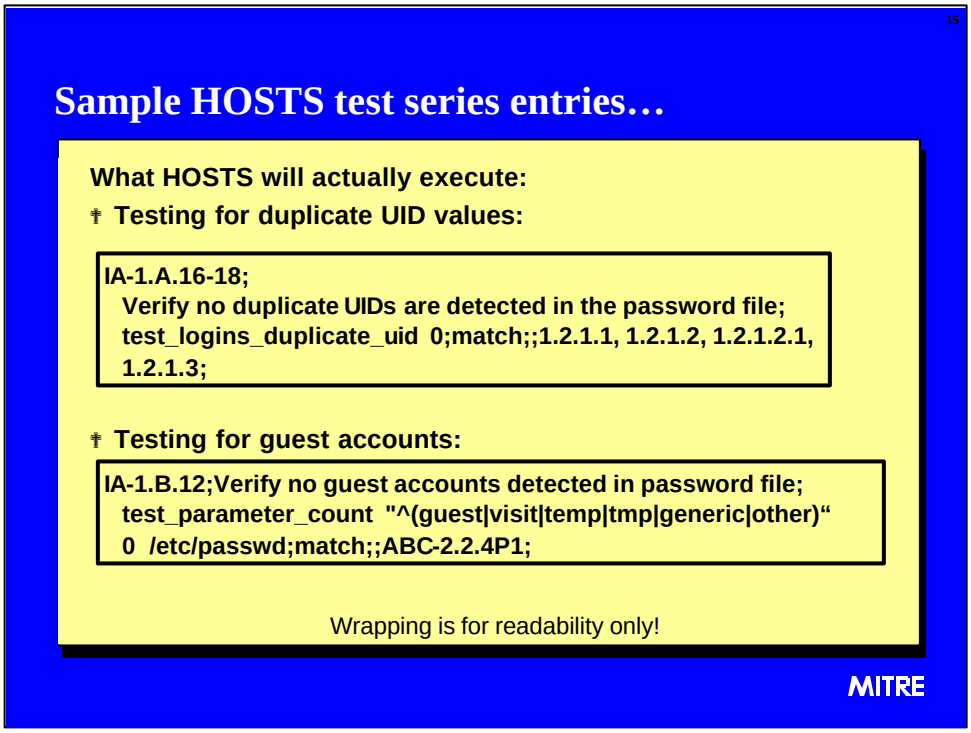


As before, each bullet shows a progressively more secure lockdown. Just as under Linux, this system could never achieve a 100% test pass rate. The following list details the same primary causes for failures as they apply to the tested system:

- The tested system needed several services (e.g., the CDE window manager).
- A few of the failures are related to the inability of the operating system to pass some of the tests. For example, Solaris has the same problem with the POSIX useradd command as does Linux. Other failures in this class include a SetUID shell script and the inability of POSIX administrative commands to be audited under the administrative class.
- As with Linux, a few of the failures are related to the approach used to access specific privilege elevating commands (e.g., ps and su) – particularly if the approaches happen to be mutually exclusive.

Caveat: As a system is tightened, it becomes “less usable” (e.g., services are being disabled). The key is to strike a balance where the system has been tightened to the fullest extent possible while at the same time mission required services are available. Extending this to the testing process means that a perfect score (e.g., 100% pass rate) is not really desirable, let alone achievable, and may reflect a state that is unusable to the system’s targeted user community.

For most commonly used applications (and services), configuration options exist that permit one to tighten the security profile of application. This facilitates the both the locking down of the application and its use – often invisibly to the user community.



Wrapping has been included for improved readability. In reality, all lines in a given box are a contiguous single line.

Guest account definition as shown is a regular expression. To enlarge (e.g., look for other variants of the word “guest”), simply add new patterns separated by the pipe symbol (|).

The corresponding HOSTS test results…

```
Failed Tests:     None

<-------------- Requirements Summary ---------------->
    Executed        Passed (100%)        Failed (  0%)
 ================================================== 
       2                 2                    0

  Req. Status    Requirement List
  ============== ==================================
  Met:           1.2.1.1, 1.2.1.2, 1.2.1.2.1, 1.2.1.3,
                 ABC-2.2.4P1
  Partially Met: Not Applicable
  Not Met:       None

       Met:            Requirements where ALL tests
                       associated with the specified
                       requirement PASSED.
       Partially Met:  Requirements where ONE OR MORE
                       tests associated with the
                       specified requirement FAILED.
       Not Met:        Requirements where ALL tests
                       associated with the specified
                       requirement FAILED.
```

MITRE

The fact that no tests failed is clearly highlighted. In addition, all tested requirements were passed. (The italics fields in the example above represent the key elements.)

Note: Some white space removal and line wrapping was done to improve readability. The results are designed to print on standard paper with a width of less than 80 columns/characters.

Examples with failed test results are included at the end of this presentation.

Future

What changes are coming to HOSTS?

MITRE

Where is HOSTS going?

- Center for Internet Security Benchmark Scoring Tool
  - Helping to define the next generation scoring tool
    - Reconfigurable
    - Adaptable
    - Integrated References
      - Multiple levels of expertise
- HOSTS
  - Authoritative references integration (Red Hat Linux)
  - New OS Releases (Solaris 9, Red Hat Linux 8)
  - Abbreviated output and reference subsection
  - Inclusion of a Java-based front end

MITRE

Questions & Comments

James A. Finegan

jfinegan@mitre.org

G025 / MS W423

7515 Colshire Drive

McLean, VA 22102

http://www.openchannelfoundation.org/projects/HOSTS

MITRE